Master Chief
Taylor Lusk
Master Chief
20 April 2009

Course: EEL 5666C
Instructors: Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz
TAs: Mike Pridgen
Thomas Vermer

Table of Contents

1 Abstract

Master Chief will roam about an arena searching for his target. Once he locates the target he will fire on it with his weapon.

2 Executive Summary

Master Chief is a predator robot. He is designed to search an area for a target and when he finds it he is programmed to terminate the target. He will use a custom designed launcher that fires Nerf darts. This is a very practical objective. The Army, among other groups, is developing similar robotic vehicles to allow soldiers to be in a hostile area without being in danger. Master Chief does exactly that. He moves by himself about his arena searching for targets. No intervention is required. To add some excitement to the project a second robot was created to duel with Master Chief.

3 Introduction

The robot platform was built in two stages: the chassis and the turret. The chassis was designed around a set of tank tracks that were purchased early in the process. This choice will be explained in Section 5. The chassis needed to be large enough and robust enough to support whatever we mounted on it. It also supports all of the obstacle avoidance equipment.

The main design point of the turret was the launcher mechanism. The size of the launcher wheels and motors, as also described in Section 5, determined the size of the turret. Another key idea was building in the ability to freely rotate about the vertical axis.

The last part of the project was writing software. Behaviors were programmed as described in Section 8. The software was tested and the whole project was put together for demonstration.

4 Integrated System

The system begins with the microcontroller. An AtMega 128 microcontroller built onto a board based on the Mavric II-B was used. This board was built by Pridgen-Vermeer Robotics and has several improvements over the Mavric. One of the major improvements is the addition of built-in regulators for the servo ports. This greatly simplified the implementation of hobby servos.

For obstacle avoidance ten bump sensors and four infrared (IR) rangefinders were used. The bump sensors were connected directly to an analog-to-digital (A/D) port on the controller board and the IR sensors were connected to the board through an A/D multiplexer. This allowed the implementation of more analog sensors than there are ports on the controller board.

As mentioned above, standard hobby servos were used. Except for the dart launcher motors, all of the actuation is done by these servos. Their versatility and ease of use made them ideal for this project.

Finally the eye of the robot is a CMU camera mounted on the front of the turret. It is mounted so that it can rotate with the turret while the robot looks for its target.

5 Mobile Platform

The platform begins with a solid chassis. This was constructed out of 1/8 in plywood held together by metal brackets. All of the plywood pieces were laminated together, created 1/4 in pieces. This yielded significantly higher strength of the platform. Tank tracks were chosen instead of wheels. Tracks are more stable and can turn in place, unlike a four-wheel system. A system of two wheels with a castor was considered but it was determined that such a system would not be stable enough to support the turret. The tracks work very well on hard surfaces but on surfaces such as carpet they tend to slip off of their guide wheels. The tracks are driven by two hacked servos. The main battery pack is mounted under the top surface and the controller board is mounted on top toward the back. Figure 1 shows the chassis configuration.

Figure 1: Chassis.

The second segment of the platform is the turret. The size of the launcher assembly was the major design point for the turret. Initially wheels of 3.5 in diameter were used. These were tested and produced excellent results. When run on two standard AA batteries the assembly launched a dart approximately 20 ft with only a few inches of drop in the

trajectory. With this success it was decided that the overall size could be reduced by using smaller wheels at a higher voltage. After two more iterations a wheel diameter of 1.5 in was selected. These are run at 4.8 V and produced excellent dart velocity and flight characteristics.

With that component completed supports were designed for the motors. Also the camera was mounted on the front. The motors are powered by a dedicated battery pack. This pack is mounted on the front, opposite the vertical axis about which the turret rotates. With this placement the batteries were able act as a counterweight against the moment created by the launcher motors. To further help support the turret several pieces were made out of PVC to help support the middle of the structure. These can be seen in Figure 2. Also six 10 mm blue LEDs were placed around the perimeter of the turret to act as a target for Master Chief's enemy to see. Though it is not usual for a robot to want his enemy to be able to see him it is necessary to maintain a fair fight so that Master Chief is challenged.

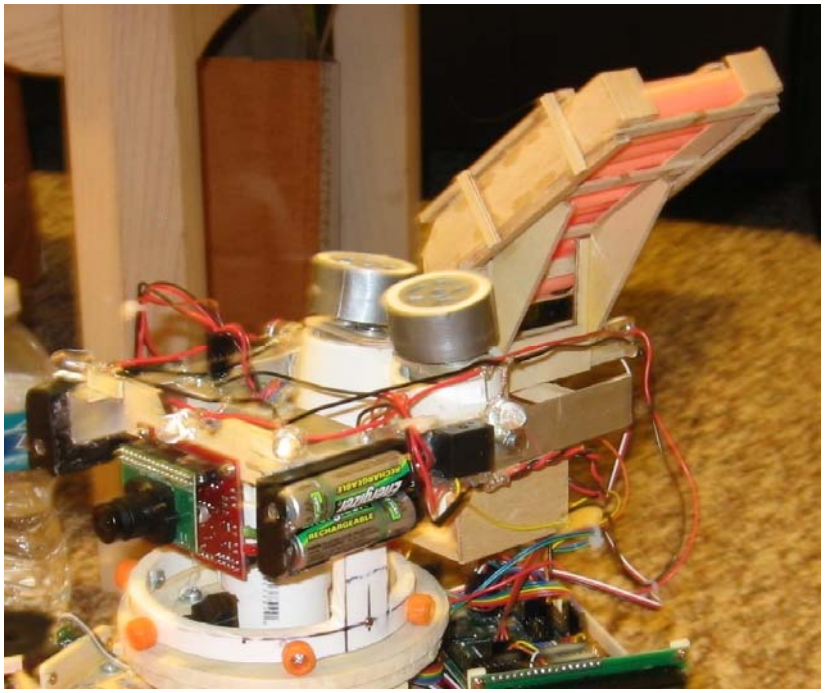

Figure 2: Turret.

A slip ring was designed and built to allow the turret to rotate freely. This would allow the electronics on the turret to be attached to the board without the problem of wires getting wrapped somewhere. During testing the assembly performed well across most of its contacts; however, the rings of the slip ring were building up capacitance which was

interfering with the high-frequency camera signal. Since this wire had to be connected directly to the board the slip ring became a purely structural element. It will be implemented on a future project.

6 Actuation

As previously stated, standard hobby servos were used for all of the actuation except propelling the darts. Two hacked servos were used as drive wheels. They were somewhat slower than hoped for but it turned out well since the robots were easier to track at the slow speed. Another servo, this time un-hacked, spins the turret. It is directly attached to the slip ring shaft. The fourth servo is used to load darts into the launcher.

The launcher motors were 12 V DC motors driven at 4.8 V. The motors are controlled by relays receiving a signal from a digital I/O pin on the controller board. These motors drove 1.5 in wheels that propelled the darts at excellent velocity. A value was never determined for the velocity but at a range of 10-15 ft the darts flew with only 1-2 in drop in trajectory. The wheels were mounted with a 7.5° offset from the horizontal plane. This feature put spin on the dart, giving it stability in flight.

7 Sensors

Master Chief employs three kinds of sensors, including infrared range finders and bump sensors. The third is the special sensor, a CMU camera. The range finders and bump sensors are used for obstacle avoidance while the CMU camera is used for target tracking. A CdS cell was going to be used to measure turret revolutions. Due to the shortcomings of the slip ring, however, this became unnecessary.

7.1 Infrared Range Finder Array

The infrared module consists of an IR LED and an IR transistor. The transistor has a specialized lens over it so that the light is refracted according to the angle at which it enters. This allows the module to determine how far from the front of the module the object sits.

Four infrared modules are mounted on the platform. One is placed on each front corner, one on the front center, and one on the back center. The front modules were visible in Figure 1. The front and back center modules point straight along the forward-backward axis. The units on the corners are angled outward to cover each side of the robot's path.

The modules are connected to an A/D channel through an A/D multiplexer. This configuration allows up to sixteen analog sensors to be connected to one port on the controller board. Each sensor returns a dimensionless value based on the voltage returned. Figures 3 and 4 show the values returned at various distances.

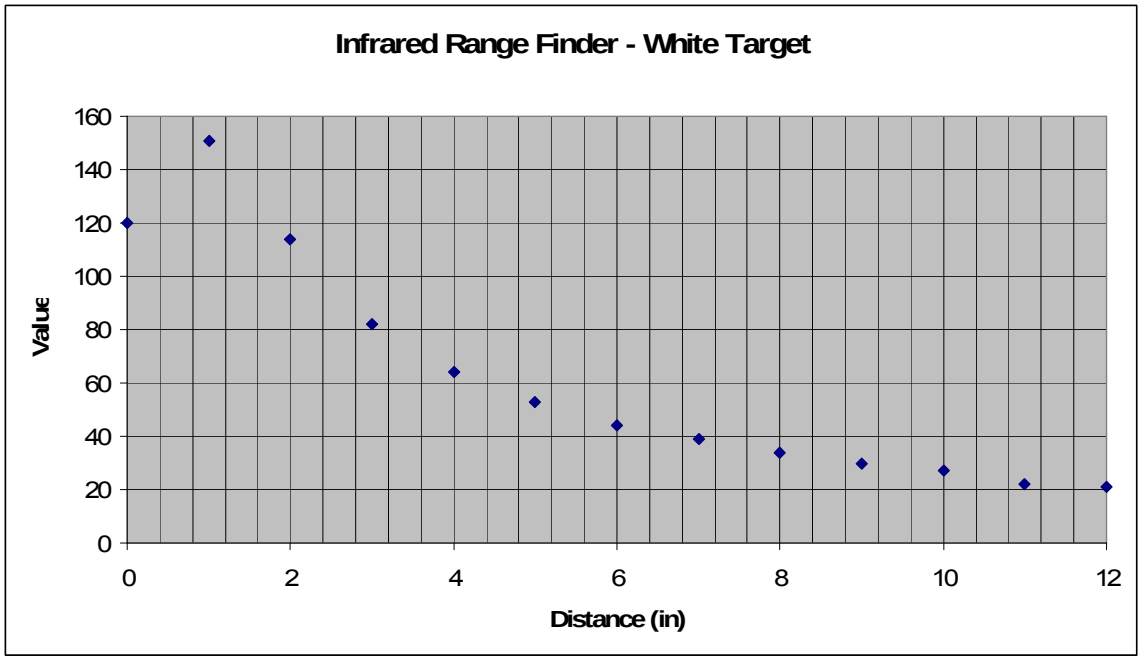

Figure 3: Infrared Range Finder calibration data for a white paper target.

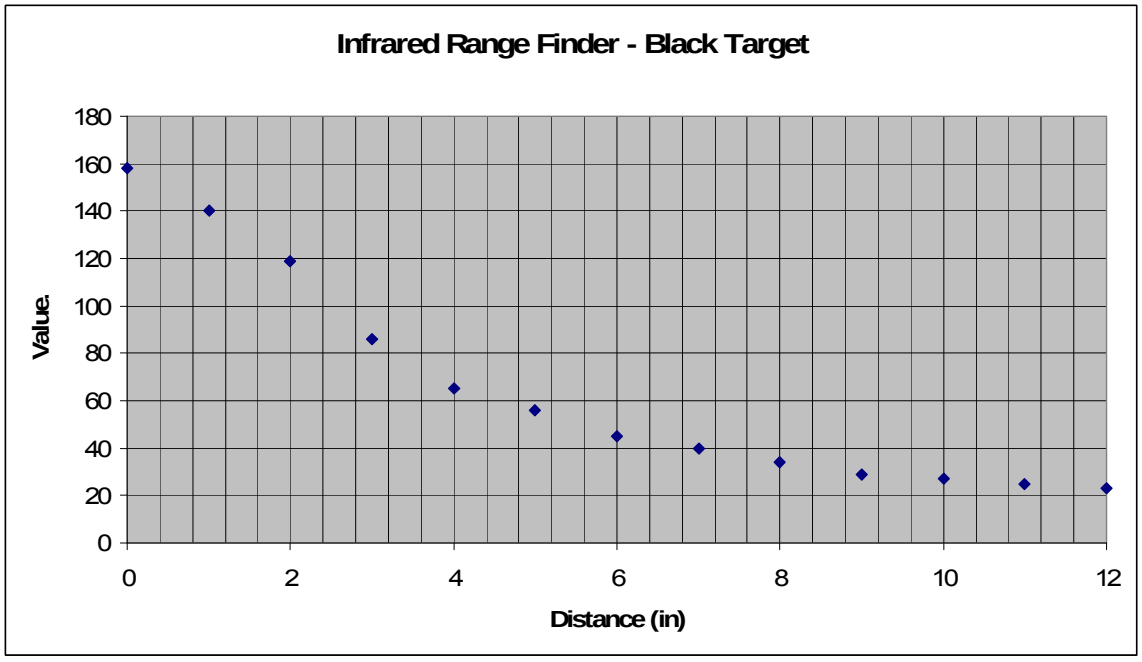

Figure 4: Infrared Range Finder calibration data for a black paper target.

The figures show that the modules are accurate from 1 in to at least 12 in. The data was taken by pointing the module first at a piece of white paper, shown in Figure 1, and then at a piece of black paper, shown in Figure 2. As expected the white paper reflected more light and therefore returned a higher value at each distance.

The modules are used for the obstacle avoidance routine. The robot turns proportionally to the value returned by the sensors.

7.2 Bump Sensor Array

Ten bump sensors are arranged around the perimeter of the platform. These sensors are simply push-button switches. The sensors are each mounted with a resistor and then wired with an eleventh resistor into a voltage divider circuit. This circuit can be seen in the Appendix in Figure A-1. The circuit in Figure A-1 is laid out to match the actual layout on the platform. Using the voltage divider allows the whole array to be read by a single A/D channel. The resistor values were selected to return a unique voltage for each sensor combination under consideration, as shown in Tables 1 and 2.

	R (Ω)	Vi	Vo	Out
R1	2000	5	2.87	147
R2	1500	5	3.21	165
R3	1000	5	3.65	187
R4	2967	5	2.38	122
R5	2700	5	2.50	128
R6	5600	5	1.63	83
R7	3300	5	2.25	115
R8	4700	5	1.82	93
R9	2200	5	2.76	141
R10	5100	5	1.73	89
R11	2700			

Table 1: Bump Sensor Array resistor values.

As shown in Table 1, there are two non-standard resistor values. R3 is 1 kΩ and R4 is 2967 Ω. R3 is made up of two 2 kΩ resistors arranged in parallel and R4 consists of three resistors-of 2.7 kΩ, 220Ω, and 47 Ω-arranged in series.

Front	Sensor ID	R (Ω)	Vi	Vo	Out
L	1, 2	857.1	5	3.80	194
R	2, 3	600.0	5	4.09	209
Split	3, 1	666.7	5	4.01	205
All	1, 2, 3	461.5	5	4.27	219

Back	Sensor ID	R (Ω)	Vi	Vo	Out
R	6, 7	2076	5	2.83	145
L	7, 8	1939	5	2.91	149
Split	8, 6	2555	5	2.57	132
All	6, 7, 8	1440	5	3.26	167

Corner	Sensor ID	R (Ω)	Vi	Vo	Out
F/R	3, 4	747.9	5	3.92	200
B/R	5, 6	1822	5	2.99	153
B/L	8, 9	1499	5	3.22	165
F/R	10, 1	1437	5	3.26	167

Diagonal	Sensor ID	R (Ω)	Vi	Vo	Out
Pos	4, 9	1212	5	3.45	177
Neg	5, 10	1876	5	2.95	151

Table 2: Bump Sensor combinations.

7.3 CMU Camera

The CMU Camera consists of a CMOS camera built onto a board controlled by a SX28 microcontroller. The CMU unit is connected to the robot controller board through a RS-232 serial port. The advantage of using a CMU Camera over using only a CMOS camera is the functionality that is built into the CMU unit. The unit comes with many useful functions pre-programmed on the SX28 to process the image received by the camera. This cuts down on the load on the main microcontroller by reducing the size of the main program, allowing the whole system to run faster. The CMU Camera board also has a servo port built in. This can be used to directly drive a servo for tracking a blob.

For Master Chief the CMU Camera is used to locate the target using color blob detection. There is a built-in function in the CMU Camera's memory to return the coordinates in the camera's view range of the centroid of the blob. This data is used for target tracking. The distance and direction of the centroid from the center of the view determines which direction and how many degrees the camera must be moved. The camera is mounted on a turret that rotates about one axis, specifically the vertical axis running through the servo used to drive the turret.

The other data from the camera that could be used is the size of the color blob. The size of the blob in the camera's view can be used to determine how far away the object is. This gives the distance to the target which could be used to adjust the elevation of a dart launcher. In the interest of simplicity this feature was not implemented.

8 Behaviors

Mater Chief employs three general behaviors. Obstacle avoidance and searching for the target run concurrently. Target termination is the third.

Obstacle avoidance is exactly that: using the infrared and bump sensors described in Section 7 Master Chief can identify obstacles and adjust his movement to stay away from them. Since Master Chief is a soldier he has other things going on as well so his obstacle avoidance is relatively simple. If he sees an object on a front side sensor he rotates his drive motors opposite each other to spin in place about 45° away from the object. If he sees an object directly in front of him he backs up a few inches and then spins. If he detects that he is in a corner he will back up and spin around 135° to get out of the corner. The bump sensors are set up in a similar fashion.

While Master Chief is moving and avoiding obstacles he is also searching for his target. The turret rotates back and forth in 5° increments and takes a picture with his CMU camera at each location. If the target color shows up in the image he stops everything and begins target termination. If the target color is not visible he continues obstacle avoiding and searching for his target.

Once the target is found Master Chief must terminate the target. He uses the x-coordinate of the centroid of the target color blob to center the turret on the target. Once he is locked on he begins his firing sequence. The launcher motors are spun up and the loader servo is actuated to launch a dart at the target. Master Chief performs well firing a shot every two seconds but he could fire faster if needed.

9 Experimental Layout and Results

Three sets of experiments were run for Master Chief: launcher performance, infrared sensor performance, and CMU camera performance.

9.1 Launcher Performance

The first set of experiments that were run was on the launcher assembly. The launcher motors were set their supports and mounted to a 2 in x 4 in block for stability. The launcher wheels were mounted to them and batteries were attached. The first run was with 9.6 V battery and 3.5 in wheels. This configuration vibrated the test stand apart within two or three seconds. The battery was changed to 7.2 V. This was also producing too much vibration so the experiment was stopped before any damage was done. Finally the voltage was dropped to 3 V. This still spun the motors at excellent RPM but the vibration at the voltage was dramatically lower than at the other voltages. Test firings were conducted at this voltage. The darts traveled 20 ft with only 3-4 in drop in

trajectory. The shots were also reasonable accurate. The launcher shot a 10 in spread at 20 ft.

Since there was so much available voltage and the assembly had such a large footprint the assembly was modified. The voltage was raised to 4.8 V and the wheels were changed for 2.5 in wheels. This configuration produced excellent dart velocity and moderate vibration.

The wheels were changed again, this time for 1.5 in wheels. These were first run at 7.2 V. By this time the robot construction was at the point where the launcher assembly could be mounted on the robot. This would give a better idea of the effects the vibration would have on the robot as a whole. This configuration produced dart velocity that almost knocked over the target, which was a full water bottle. The vibration was high enough to be troublesome. The darts in the hopper were shaken around enough to cause two misfires in each of two nine-shot tests. The darts themselves were taking damage, indicated by the orange streaks on the launcher wheels. Also the CMOS camera board was shaken all the way out of its socket on the CMU board. Since there was excess velocity and the robot was sustaining damage from the vibration the motor voltage was lowered to 4.8 V. This still produce plenty of shot velocity while greatly reducing vibration. This configuration became the final configuration.

9.2 Infrared Sensor Performance

A curve needed to be generated to determine what voltage the IR modules returned at what range. The data sheet had such a curve but that was for best case conditions. The test consisted of simply pointing the module at a surface while measuring the distance and taking the output. Two tests were run for this section, one with a white paper target and one with a black paper target. The results were presented in Figures 3 and 4.

9.3 CMU Camera Performance

A simple test was performed with the camera to determine what values certain colors produced. This test was run under a 60 W kitchen light for consistency. The targets were pieces of poster board that were brightly colored. Green, yellow, pink, and two shades of orange were used. The CMU command simply returns the mean color in the view area. The results are shown in Table 3.

	Means			Deviations		
	Red	Green	Blue	Red	Green	Blue
Green	83.4	146	47.8	5	7	3
Yellow	160.4	107	16	10	5.6	0
Lt Orange	150.6	39	16	10	2	0
Dk Orange	158.6	16	16	8.4	0	0
Pink	156.4	16	16	10	0	0

Table 3: Color data from CMU camera.

10 Conclusion

The platform and turret construction turned out very well. Only minor changes to the design had to be made while building. Further, the robot is very sturdy except in one or two places.

The robot performed generally as designed. It avoided obstacles well while searching for its target. Once the target was found, however, Master Chief only liked to track the target on some runs. For the rest of the runs he just stopped his turret when he saw the target. When he did track the target he was able to effectively lock on. He shot at and hit his target almost every time. When he just stopped his turret he obviously had more difficulty. If the target was stationary it was a guaranteed miss. This is because the turret stopped when it saw the target instead of when it was centered on the target.

The software can be improved in many aspects. Because of poor time management and construction setbacks insufficient time was allotted for programming. Also some modification to the slip ring can be designed so that it can operate all of the devices on the turret. That would allow for full rotation of the turret and therefore more versatility of the robot. Finally, motor drivers were obtained and built for the launcher motors but they were not used due to their complexity. Implementing these would further enhance the versatility of the robot.

11 Documentation

Datasheets and class lectures were the only documentation used for this project. Code was contributed from several people. Pridgen and Vermeer Robotics contributed servo code, LCD code, A/D code, and delay code for direct use. Dana Preble, Joe Bari, and Andrei Kamalov all contributed code that was referenced for the CMU camera. The UART Library written by Peter Fleury was also used to control the camera.

12 Appendices

12.1 Circuit Diagrams

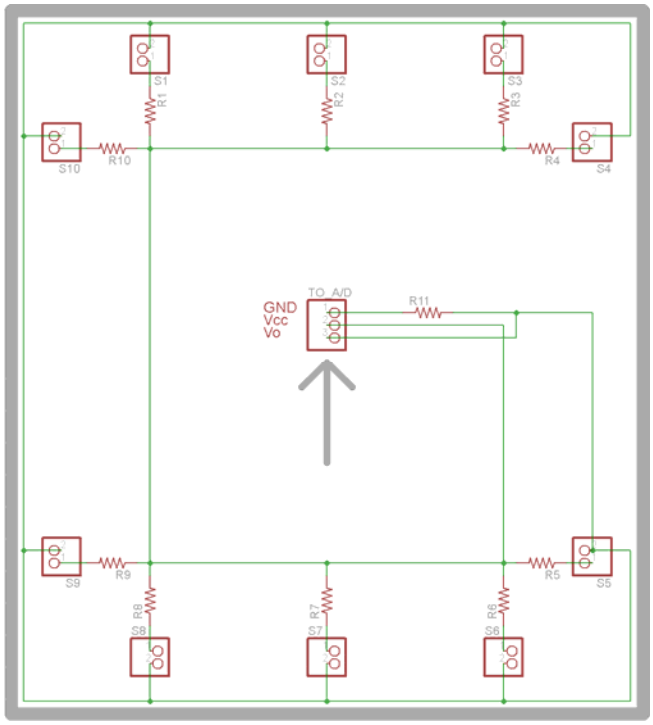

Figure A-1: Bump Sensor Array diagram.

12.2 CAD Drawings

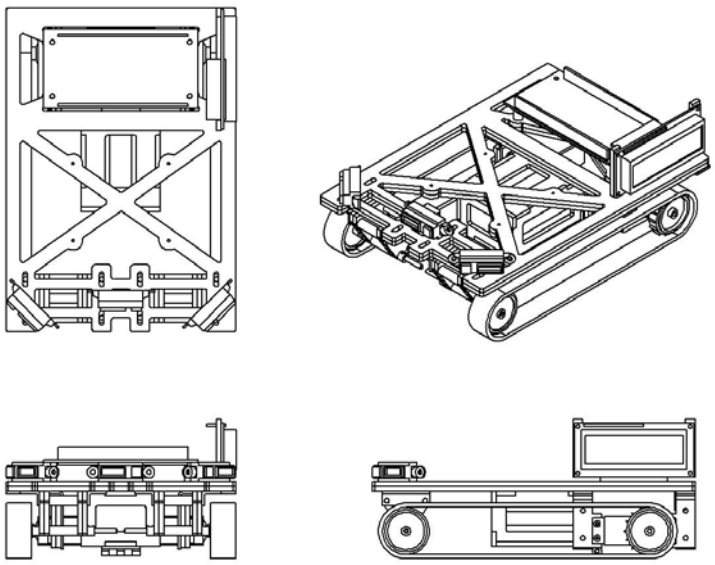

Figure A-2: Chassis

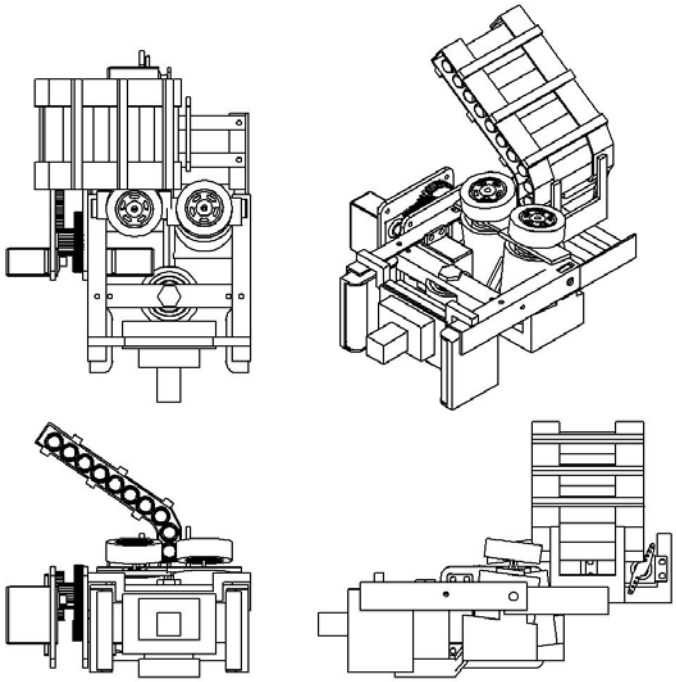

Figure A-3: Turret

12.3 Code

Only original programs are included here. The following programs and header files were used from Pridgen-Vermeer:

ADC.c
LCD.c
pvr_servos.c
sleep.c
UART.c
ADC.h
LCD.h
PVR_Servos.h
sleep.h

These files are available on request from Pridgen-Vermeer. The following program and header file were used from the UART Library by Fleury:

UART.c
uart.h

A description of the UART Library and a link to download the files can be found at:

http://homepage.hispeed.ch/peterfleury/group__pfleury__uart.html

12.3.1 MasterChief.c

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "PVR_Servos.h"
#include "sleep.h"
#include "LCD.h"
#include "ADC.h"
#include "uart.h"

#ifndef F_CPU
#define F_CPU 16000000UL      // Atmega128 runs at 16mHz
#endif
#define UART_BAUD_RATE 19200   // Baud rate

void receive(void);
void shoot(void);
void track(void);

int data[8], pix, con, shotCnt = 0, increment = 0;
int found = 0, lock = 0, count = 0, TC = 0;

int main(void)
{
    int i, j = 0, bump = 0, IR[4], p = 0b00000000, up = 1, found = 0;
    int loc[4] = {0, 0, 0, 0};
    char c = '\n';

    // Initialization functions
    initSleep();           // Sleep
    lcdInit();             // LCD display
    initServo();           // Servos
    initADC();             // A/D channels
    DDRA = 0b00001111;     // Port A (for A/D Mux) - First four bits to ouput
    ms_sleep(100);
    uart_init( UART_BAUD_SELECT( UART_BAUD_RATE, F_CPU ) );
    sei();
    lcdString("Lusk Robotics");

    lcdClear();
    uart_puts("PM 1 \r");
    ms_sleep(500);
    do {
        c = uart_getc();
    } while (c != ':');
    ms_sleep(1000);

    lcdClear();
    uart_puts("SW 1 1 80 143 \r");
    ms_sleep(500);
    do {
        c = uart_getc();
    } while (c != ':');
    ms_sleep(1000);

    lcdClear();
    uart_puts("MM 1 \r");
    ms_sleep(500);
    do {
        c = uart_getc();
    } while (c != ':');
    ms_sleep(1000);
    // End initialization

    moveServo4(-70);

    while(shotCnt < 10)
    {
        moveServo1(-100);
        moveServo2(100);
        moveServo3(increment);
```

```
        ms_sleep(50);
        for (i = 0; i < 4; i++)
        {
            PORTA = p;
            p = p + 2;
            IR[i] = adcOne();
            if (IR[i] > 80) loc[i] = 1;
        }
        if (IR[0] > 80 || IR[1] > 80 || IR[2] > 80 || IR[3] > 80)
        {
            moveServo1(0);
            moveServo2(0);
            moveServo3(0);
            ms_sleep(25);
            IRAvoid(loc, IR);
            loc[0] = 0;
            loc[1] = 0;
            loc[2] = 0;
            loc[3] = 0;
        }
        bump = adcZero();
        if (bump > 20)
        {
            moveServo1(0);
            moveServo2(0);
            moveServo3(0);
            ms_sleep(100);
            bumpAvoid(bump);
            lcdClear();
        }
        if (!TC) uart_puts("TC 155 255 0 32 0 32 \r");
        ms_sleep(50);

        receive();
        if (data[0] > 0)
        {
            found = 1;
            moveServo1(0);
            moveServo2(0);
            lcdClear();
            lcdString("Found");
            ms_sleep(100);
            while (found && shotCnt < 5)
            {
    //          track();
                shoot();
            }
        }
        if (increment < 100 && up == 1 && !found) increment = increment + 10;
        if (increment > -100 && up == 0 && !found) increment = increment - 10;
        if (increment > 99) up = 0;
        if (increment < -99) up = 1;
    }
    return 0;
}

void receive(void)
{
    int i, x = 1;
    char temp[3], c = '\n';
    if (!TC)
    {
        while (c != 'A' && c != 'N') c = uart_getc();
        if (c == 'N') {
            while (c != ':') c = uart_getc();
            lcdString("Command failed");
            ms_sleep(1000);
            lcdClear();
        }
        TC = 1;
    }
```

```
    do {
        c = uart_getc();
        lcdChar(c);
    } while (c != 'M');
    c = uart_getc();
    i = 0;
    temp[0] = uart_getc();
    temp[1] = uart_getc();
    if ((temp[1] >= 0x30) && (temp[1] <= 0x39))
    {
        x = 2;
        temp[2] = uart_getc();
        if (temp[2] >= 0x30 && temp[2] <= 0x39)
        {
            data[0] = 100 * (temp[0] - 0x30) + 10 * (temp[1] - 0x30) + (temp[2] - 0x30);
        }
        else data[0] = 10 * (temp[0] - 0x30) + (temp[1] - 0x30);
    }
    else data[0] = temp[0] - 0x30;
    lcdInt(data[0]);
    ms_sleep(250);
    lcdClear();
}

/*void track(void)
{
    if (data[0] > 32 && data[0] < 48) count++;       // Center turret on x = 40
    else count = 0;
    if (count == 5) lock = 1; // If held on target for ~4 seconds lock = true
    if (!lock)
    {
        if (data[0] < 36) increment = increment + 5;
        if (data[0] > 44) increment = increment - 5;
        moveServo3(increment);
        ms_sleep(50);
        receive();
//      if ((data[0] < 10) || (data[0] > 70)) found = 0;     // If color lost return to
search loop
        if (data[0] == 0) found = 0;
    }
    if (lock && shotCnt < 10)
    {
        lcdClear();
        lcdString("Lock");
        shoot();
        receive();
        if (data[0] < 10 || data[0] > 70)
        {
            found = 0;
            lock = 0;
        }
        if (data[0] < 33 || data[0] > 47) lock = 0;
        if (data[0] < 36) increment = increment + 2.5;
        if (data[0] > 44) increment = increment - 2.5;
        moveServo3(increment);
        ms_sleep(50);
    }

}*/

void shoot(void)
{
    DDRE = 0b11111111;
    shotCnt++;
    PORTE = 0b11100000;
    moveServo4(80);
    ms_sleep(1000);
    PORTE = 0b00000000;
    moveServo4(-70);
    ms_sleep(1000);
    DDRE = 0b00100000;
```

```
}
```

12.3.2 Avoid.c

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "PVR_Servos.h"
#include "sleep.h"
#include "LCD.h"
#include "ADC.h"

void IRAvoid(int loc[], int IR[]);
void bumpAvoid(int value);

void IRAvoid(int loc[], int IR[])
{
    int time = 1000, done = 0, back = 0, bump, i, corners = 0;

    // Front Left
    if (loc[0] == 1)
    {
        if (IR[2] > 60)  // Check if robot is in a corner
        {
            lcdString("Corner");
            for (i = 0; i < 20 && back < 80; i++) // Check rear IR
            {
                moveServo1(100);
                moveServo2(-100);
                ms_sleep(100);
                PORTA = 0b00000110;
                back = adcOne();
            }
            moveServo1(0);
            moveServo2(0);
            ms_sleep(25);
            if (IR[0] > IR[2])
            {
                moveServo1(100);
                moveServo2(100);
            }
            else
            {
                moveServo1(-100);
                moveServo2(-100);
            }
            for (i = 0; i < 25 && !corners; i++)
            {
                ms_sleep(100);
                bump = adcZero();
                if (bump > 20) corners = 1;
            }
            if (corners) bumpAvoid(bump);
            lcdClear();
            done = 1;
        }
        else
        {
            lcdString("1");
            moveServo1(100);
            moveServo2(100);
            ms_sleep(time);
            lcdClear();
            done = 1;
        }
    }

    // Front Center
    if (loc[1] == 1 && done == 0)
    {
        lcdString("2");
        for (i = 0; i < 20 && back < 80; i++) // Check rear IR
```

```
    {
        moveServo1(100);
        moveServo2(-100);
        ms_sleep(100);
        PORTA = 0b00000110;
        back = adcOne();
    }
    moveServo1(0);
    moveServo2(0);
    ms_sleep(50);
    if (IR[0] > IR[2])
    {
        moveServo1(100);
        moveServo2(100);
    }
    else
    {
        moveServo1(-100);
        moveServo2(-100);
    }
    ms_sleep(time);
    lcdClear();
    done = 1;
}

// Front Right
if (loc[2] == 1 && done == 0)
{
    if (IR[0] > 60)  // Check if robot is in a corner
    {
        lcdString("Corner");
        for (i = 0; i < 20 && back < 80; i++) // Check rear IR
        {
            moveServo1(100);
            moveServo2(-100);
            ms_sleep(100);
            PORTA = 0b00000110;
            back = adcOne();
        }
        moveServo1(0);
        moveServo2(0);
        ms_sleep(25);
        if (IR[0] > IR[2])
        {
            moveServo1(100);
            moveServo2(100);
        }
        else
        {
            moveServo1(-100);
            moveServo2(-100);
        }
        for (i = 0; i < 25 && !corners; i++)
        {
            ms_sleep(100);
            bump = adcZero();
            if (bump > 20) corners = 1;
        }
        if (corners) bumpAvoid(bump);
        lcdClear();
        done = 1;
    }
    else
    {
        lcdString("3");
        moveServo1(-100);
        moveServo2(-100);
        ms_sleep(time);
        lcdClear();
        done = 1;
    }
```

```
    }

    // Rear Center
    if (loc[3] == 1 && done == 0)
    {
        lcdString("4");
        moveServo1(-100);
        moveServo2(100);
        for (i = 0; i < 10 && !corners; i++)
        {
            ms_sleep(100);
            bump = adcZero();
            if (bump > 20) corners = 1;
        }
        if (corners) bumpAvoid(bump);
        lcdClear();
        done = 1;
    }

    moveServo1(0);
    moveServo2(0);
    ms_sleep(100);
}

void bumpAvoid(int value)
{
    int time = 2000;
    // R1
    if (value < 149 && value > 146)
    {
        lcdString("1");
        moveServo1(100);
        moveServo2(-100);
        ms_sleep(time);
        moveServo1(0);
        moveServo2(0);
        ms_sleep(250);
        moveServo1(100);
        moveServo2(100);
        ms_sleep(time);
        lcdClear();
    }

    // R2
    if (value < 166 && value > 163)
    {
        lcdString("2");
        moveServo1(100);
        moveServo2(-100);
        ms_sleep(time);
        moveServo1(0);
        moveServo2(0);
        ms_sleep(250);
        moveServo1(100);
        moveServo2(100);
        ms_sleep(2*time);
        moveServo1(0);
        moveServo2(0);
        ms_sleep(250);
        lcdClear();
    }

    // R3
    if (value < 188 && value > 185)
    {
        lcdString("3");
        moveServo1(100);
        moveServo2(-100);
        ms_sleep(time);
        moveServo1(0);
        moveServo2(0);
```

```
    ms_sleep(250);
    moveServo1(-100);
    moveServo2(-100);
    ms_sleep(time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
    lcdClear();
}

// R4
if (value < 121 && value > 119)
{
    lcdString("4");
    moveServo1(-100);
    moveServo2(-100);
    ms_sleep(time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
    lcdClear();
}

// R5
if (value < 129 && value > 126)
{
    lcdString("5");
    moveServo1(100);
    moveServo2(100);
    ms_sleep(time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
    lcdClear();
}

// R6
if (value < 85 && value > 82)
{
    lcdString("6");
    moveServo1(-100);
    moveServo2(100);
    ms_sleep(time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
    moveServo1(100);
    moveServo2(100);
    ms_sleep(time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
    lcdClear();
}

// R7
if (value < 116 && value > 114)
{
    lcdString("7");
    moveServo1(-100);
    moveServo2(100);
    ms_sleep(time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
    moveServo1(100);
    moveServo2(100);
    ms_sleep(2*time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
```

```
    lcdClear();
}

// R8
if (value < 95 && value > 92)
{
    lcdString("8");
    moveServo1(-100);
    moveServo2(100);
    ms_sleep(time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
    moveServo1(-100);
    moveServo2(-100);
    ms_sleep(time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
    lcdClear();
}

// R9
if (value < 90 && value > 88)
{
    lcdString("9");
    moveServo1(-100);
    moveServo2(-100);
    ms_sleep(time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
    lcdClear();
}

// R10
if (value < 142 && value > 139)
{
    lcdString("10");
    moveServo1(100);
    moveServo2(100);
    ms_sleep(time);
    moveServo1(0);
    moveServo2(0);
    ms_sleep(250);
    lcdClear();
}
```